For large file transfers within the NAS enclave, use the commands shiftc, mcp, or cxfscp. The slower cp command also can be used.

The following file transfer commands can be used when both the source and destination locations are accessible on the same host where the command is issued.

Shift Command

Shift (shiftc) is a NAS-developed tool for performing automated local and remote file transfers. Shift utilizes a variety of underlying file transports to achieve maximum performance for files of any size on any file system.

Where is it installed at NAS?

shiftc is installed on the Pleiades and Lou front-end systems (PFEs and LFEs).

When to use it?

The command shiftc can be used as a drop-in replacement for cp at any time on any system on which it is installed.

Examples

lfe*5*% shiftc /nobackup/*username/filename* $HOME
pfe*27*% shiftc $HOME/*filename* /nobackup/*username*

Performance

shiftc utilizes mcp, when available, so it can be up to 10 times faster than cp for large files (2+ GB) and can achieve up to 1.8 GB/sec on a single host. Using the --hosts option, it has been measured to achieve up to 5.0 GB/sec on 8 hosts.

For more information, see Shift File Transfer Overview.

cxfscp

cxfscp is a program for quickly copying large files to and from a CXFS filesystem, such as the Lou front-end (LFE) home file system. It can be significantly faster than cp on CXFS filesystems since it uses multiple threads and large direct I/Os to fully utilize the bandwidth to the storage hardware.

For files less than 64 kilobytes in size, which will not benefit from large direct I/Os, cxfscp will use a separate thread for copying these files using buffered I/O similar to cp.

Where is it installed at NAS?

cxfscp is installed on the LFEs.

When to use it?

The Pleiades Lustre filesystems (/nobackupp*X*) are mounted on lfe[*5-8*]. The command cxfscp can be issued on either of these hosts to copy large files between the LFE's CXFS home file system and Pleiades's /nobackup. This is an easy way to transfer files between Lou and Pleiades without the need for scp or scp.

Examples

lfe*5*% cxfscp -rp --bi /nobackupp4/*username*/*testdir*_a /u/*username*/tests
lfe*5*% cxfscp -p --bo /u/*username*/data* /nobackupp4/*username*/*data_dir*

Performance

Some benchmarks done by NAS staff show that cxfscp is typically 4-7 times faster than cp for large files (2+ GB) and can achieve up to 650 MB/sec.

For more information, read **man cxfscp**.

cp

cp is a Unix command for copying files between two locations (for example, two different directories of the same filesystem or two different filesystems such as NFS, CXFS or Lustre).

Where is it installed at NAS?

cp is available on all NAS systems except the secure front ends (SFEs).

Use the -p option to preserve the timestamp on the file.

Examples

pfe*21*% cp -p $HOME/*filename* $HOME/*newdir/filename2*
pfe*21*% cp -p $HOME/*filename* /nobackup/*username*

The NAS-developed Shift tool can copy files locally on NAS enclave hosts, transfer files between hosts inside the NAS enclave, and transfer files between the NAS enclave and remote hosts. You can also use Shift to check the status of transfers at any time, receive email notification of completion, errors, and warnings, and restart interrupted transfers or transfers with errors.

All functionality is accessed through the Shift client, which is invoked via the `shiftc` command. The syntax for `shiftc` is similar to the syntax for the `cp` and `scp` commands.

Shift is the recommended method for transferring files to and from the Lou mass storage system, as it can create tar files as part of the transfer and split a transfer into multiple tar files for oversized directories (larger than 1 TB).

Advanced Features

Shift includes the following advanced features:

- Automatic parallelization of transfers
- Local and remote tar creation and extraction
- Synchronization based on modification times and checksums (similar to `rsync`)
- Automatic file integrity verification and correction
- Ability to restart transfers
- Automatic retrieval of files from tape storage (DMF-managed Lou filesystems)
- Ability to check status of current transfers

How to Use Shift

See the following articles for detailed information about how to use Shift:

- Using Shift for Local Transfers and Tar Operations
- Using Shift for Transfers and Tar Operations Between Two NAS Hosts
- Using Shift for Remote Transfers and Tar Operations
- Checking Shift Transfer Status and Restarting Transfers
- Shift Command Options

Additional Resources

You can also see presentation slides for three HECC training webinars that demonstrate how to use the tool:

- Simplifying and Optimizing Your Data Transfers (PDF)
- Advanced Features of the Shift Automated File Transfer Tool (PDF)
- Simple Automated File Transfers Using SUP and Shift (PDF)

Recordings of each presentation are also available in the Past Webinars Archive.

Note: Some hostnames and options may have changed since the webinars were presented.

Writing Data to Pleiades Filesystems

A common way to share files and/or directories with group members or others is to use the `chmod` command to change the permissions. However, `chmod` has limitations, so you may sometimes choose to use Access Control Lists (ACLs).

When you issue the command `chmod g+rx` *filename*, for example, all the members in your group (`g`) gain read (`r`) and search/execute (`x`) access to that file, as shown below:

```
% ls -l filename
-rw------- 1 zsmith s0101 9 Jun 10 12:11 filename

% chmod g+rx filename

% ls -l filename
-rw-r-x--- 1 zsmith s0101 9 Jun 10 12:11 filename
```

However, `chmod` does not allow you to select which members of your group or which specific individuals outside of your group can access your files/directories. ACLs provide a mechanism for greater control of file sharing. There are two ACL commands:

setfacl

Set file access control lists.

```
SYNOPSIS
     setfacl [-bkndRLPvh] [{-m|-x} acl_spec] [{-M|-X} acl_file] file ...

     setfacl --restore=file
```

A detailed usage explanation of `setfacl` and its options can be found via **man setfacl**. Among the options listed:

1. The `-m` or `-M` option lets you "modify" the ACL, where `-m` expects an ACL on the command line and `-M` expects an ACL from a file or from standard input
2. The `-x` or `-X` option removes the ACL entries
3. The `-R` or `--recursive` option applies operations to all files and directories recursively
4. The `--test` option allows you to test the effect of changing the ACL without actually changing it
5. The `-b` option removes all extended ACL entries except the base entries of the owner, group, and others

getfacl

Get file access control lists.

```
SYNOPSIS
     getfacl [-dRLPvh] file ...

     getfacl [-dRLPvh] -
```

A detailed usage explanation of `getfacl` and its options can be found via **man getfacl**.

Note: Before you grant another user or group access to certain files or directories, make sure that access to the parent directory (where the files or directories reside) is also allowed.

Example 1

To allow another user (*jbrown*) to have read/execute (`rx`) permission on a file (*filename*) and to view the ACL before and after an ACL change:

```
% ls -l filename
-rw------- 1 zsmith s0101 9 Jun 10 12:11 filename

% getfacl filename
# file: filename
# owner: zsmith
# group: s0101
user::rw-
group::---
other::---

% setfacl -m u:jbrown:rx filename

% getfacl filename
# file: filename
# owner: zsmith
# group: s0101
user::rw-
```

user:*jbrown*:r-x
group::---
mask::r-x
other::---

% ls -l *filename*
-rw-**r-x**---**+** 1 *zsmith* s0101 9 Jun 10 12:11 *filename*

Example 2

To remove all extended ACLs in Example 1 except the base entries of the owner, group, and others:

% setfacl -b *filename*

% ls -l *filename*
-rw------- 1 *zsmith* s0101 9 Jun 10 12:11 *filename*

% getfacl *filename*
# file: *filename*
# owner: *zsmith*
# group: s0101
user::rw-
group::---
other::---

Example 3

Continuing from Example 1, to test the granting of read/execute (rx) access to another group (group id 24176) without actually doing it:

% setfacl --test -m g:24176:rx *filename*
filename: u::rw-,u:*jbrown*:r-x,g::---,**g:g24176:r-x,m::r-x**,o::---,*

% getfacl *filename*
# file: *filename*
# owner: *zsmith*
# group: s0101
user::rw-
user:*jbrown*:r-x
group::---
mask::r-x
other::---

Example 4

To allow another user (*jbrown*) recursive access to a directory (dir.abc which contains a file *filename*):

% ls -ld dir.abc
drwx------ 2 *zsmith* s0101 17 Jun 10 13:19 dir.abc

% ls -l dir.abc
total 0
-rw------- 1 *zsmith* s0101 0 Jun 10 13:19 *filename*

% setfacl -R -m u:*jbrown*:rx dir.abc

% getfacl dir.abc
# file: dir.abc
# owner: *zsmith*
# group: s0101
user::rwx
user:*jbrown*:r-x
group::---
mask::r-x
other::---

% getfacl dir.abc/*filename*
# file: dir.abc/*filename*
# owner: *zsmith*
# group: s0101
user::rw-
user:*jbrown*:r-x
group::---
mask::r-x
other::---

% ls -ld dir.abc
drwx**r-x**---**+** 2 *zsmith* s0101 17 Jun 10 13:19 dir.abc

```
% ls -l dir.abc
total 0
-rw-r-x---+ 1 zsmith s0101 0 Jun 10 13:19 filename
```

Example 5

Continuing from Example 4, to recursively remove all permissions user *jbrown* for a directory:

```
% setfacl -R -x u:jbrown dir.abc

% getfacl dir.abc
# file: dir.abc
# owner: zsmith
# group: s0101
user::rwx
group::---
mask::---
other::---

% getfacl dir.abc/filename
# file: dir.abc/filename
# owner: zsmith
# group: s0101
user::rw-
group::---
mask::---
other::---
```

For more information on ACLs, read **man acl**.

Using 'Giveto' to Copy Files and/or Directories to Another User

NAS's in-house developed giveto script is built on the use of Access Control Lists (ACLs). It allows one user (the giver) to copy files and/or directories to a /nobackup directory of another user (the recipient).

giveto is installed under /usr/local/bin on Pleiades and Lou.

In the example below, user zsmith gives a copy of his dir.abc directory on Pleiades to user jbrown. The steps describe the giveto command used by each of them, and the results.

1. User *jbrown* uses the command giveto -i *zsmith* to automatically (a) create an INCOMING directory (if it does not already exist) under her /nobackup/*jbrown* and (b) grant user *zsmith* read/write/execute permission on this directory.

   pfe*21*:/u/*jbrown*% giveto -i *zsmith*
   nobackup[*1*] = /nobackup/*jbrown*

   pfe*21*:/u/*jbrown*% ls -ld /nobackup/*jbrown*/INCOMING
   drwx**rwx**---**+** 2 *jbrown* s0202 4096 Jun 14 12:18 /nobackup/*jbrown*/INCOMING

2. User *zsmith* uses the command giveto *jbrown* dir.abc to automatically (a) create a subdirectory called *zsmith*_0 under *jbrown*'s INCOMING directory, (b) copy dir.abc to /nobackup/*jbrown*/INCOMING/*zsmith*_0, (c) grant user jbrown read/write/execute permission on /nobackup/*jbrown*/INCOMING/*zsmith*_0, and (d) send an email to user *jbrown* regarding the copy.

   pfe*21*:/home1/*zsmith*> ls -ld dir.abc
   drwx------ 2 *zsmith* s0101 17 Jun 14 12:21 dir.abc/

   pfe*21*:/home1/*zsmith*> giveto *jbrown* dir.abc
   setfacl -m u:*jbrown*:rwx *zsmith*_0
   setfacl -m u:*jbrown*:rwx *zsmith*_0/giveto.log
   setfacl -m u:*jbrown*:rwx *zsmith*_0/dir.abc
   setfacl -m u:*jbrown*:rwx *zsmith*_0/dir.abc/foo2
   path = /nobackup/*jbrown*/INCOMING/*zsmith*_0
   total 12
   drwxrwx---+ 3 *zsmith* s0101 4096 Jun 14 12:29 .
   drwxrwx---+ 3 *jbrown* s0202 4096 Jun 14 12:29 ..
   drwxrwx---+ 2 *zsmith* s0101 4096 Jun 14 12:21 dir.abc
   -rw-rwx---+ 1 *zsmith* s0101 44 Jun 14 12:29 giveto.log

   Note: If the directory *zsmith*_0 already exists prior to this step, *zsmith*_1 would be used instead.

3. User *jbrown* receives an email from user *zsmith* with a subject line "giveto files". They see that the directory dir.abc has been copied successfully. Even though the directory /nobackup/*jbrown*/INCOMING/*zsmith*_0 is still owned by user *zsmith*, user *jbrown* now has permission to read/write/execute files and directories under /nobackup/*jbrown*/INCOMING/*zsmith*_0.

   pfe*21*:/u/*jbrown*% ls -lrt /nobackup/*jbrown*/INCOMING
   total 4
   drwx**rwx**---**+** 3 *zsmith* s0101 4096 Jun 14 12:29 *zsmith*_0

   pfe*21*:/u/*jbrown*%ls -lrt /nobackup/*jbrown*/INCOMING/*zsmith*_0
   total 4
   drwx**rwx**---**+** 2 *zsmith* s0101 4096 Jun 14 12:21 dir.abc

   pfe*21*:/u/*jbrown*%ls -lrt /nobackup/*jbrown*/INCOMING/*zsmith*_0/dir.abc
   total 4
   -rw-**rwx**---**+** 1 *zsmith* s0101 8 Jun 14 12:21 *foo2*

   pfe*21*:/u/*jbrown*%getfacl /nobackup/*jbrown*/INCOMING/*zsmith*_0/dir.abc
   # file: /nobackup/*jbrown*/INCOMING/*zsmith*_0/dir.abc
   # owner: *zsmith*
   # group: s0101
   user::rwx
   user:*jbrown*:rwx
   group::---
   mask::rwx
   other::---

Read **man giveto** for more information.

The giveto script was created by NAS staff member Arthur Lazanoff.

Storing Data in Pleiades Home Filesystems

All data stored in your home filesystem of any NAS system, such as Pleiades, are automatically backed up *daily* under script control.

If you lose important data from a home filesystem and need to have it restored, please send a request to the NAS Control Room at support@nas.nasa.gov and provide the following information:

- System name
- Your username
- The directory and/or file name(s) that you wish to restore
- The date the data was last touched

Note: Disk space quota limits are imposed on the Pleiades home filesystems. If your data exceeds your disk space quota limits, reduce your usage by deleting unneeded files, copying important files to Lou, or copying them to storage space at your local site and then removing them from Pleiades.

Storing Data in Pleiades /nobackup Filesystems

Data in any /nobackup filesystem of any NAS system, such as Pleiades /nobackupp*X*, are *not* backed up by NAS. You are responsible for performing your own backups.

Note: Disk space and inode quota limits are imposed on the Pleiades Lustre filesystems (/nobackupp*X*). If your data exceeds the soft quota limits, reduce your usage by removing some files and/or archiving any important data on Lou or on the storage space at your local site.

In addition, when the overall usage of a /nobackup filesystem is near its capacity, files can be deleted by system administrators. Normally, the few top users are sent emails and asked to clean up their disk space.

Using Shift to Copy Files to Lou

The Pleiades /nobackup filesystems (for example, /nobackup[*1-8*]) are mounted on Lou. As a result, you can log into Lou and copy the files from your /nobackup filesystem to your home directory on Lou.

We recommend using the Shift tool (shiftc), which will automatically utilize the highest performing local file transport. You can also use shiftc if you need to initiate the transfer from Pleiades. If you need to encrypt the data, even within the NAS enclave, then use the shiftc --encrypt option.

A Lustre filesystem is a high-performance shared filesystem for Linux clusters that is managed with Lustre software. It is highly scalable and can support many thousands of client nodes, petabytes of storage, and hundreds of gigabytes per second of I/O throughput. The NAS Lustre filesystems are named "/nobackupp*X*."

Each Lustre filesystem is actually a set of many small filesystems, which are referred to as object storage targets (OSTs). The Lustre software presents the OSTs as a single unified filesystem.

For more information about OSTs and other Lustre filesystem components, see Main Lustre Components.

Useful Lustre Commands

The examples in this section use /nobackupp*17* as an example of a specific Lustre filesystem.

Listing Disk Usage and Quotas

To display disk usage and limits on your /nobackup directory:

pfe*21*% lfs quota -h -u *username* /nobackupp*17*

or

pfe*21*% lfs quota -h -u *username* /nobackup/*username*

To display usage on each OST, add the -v option:

pfe*21*% lfs quota -h -v -u *username* /nobackup/*username*

Listing Space Usage

To list space usage per OST and MDT, in human-readable format, for all Lustre filesystems or for a specific one:

pfe*21*% lfs df -h
pfe*21*% lfs df -h /nobackupp*17*

Listing Inode Usage

To list inode usage for all filesystems or for a specific one:

pfe*21*% lfs df -i
pfe*21*% lfs df -i /nobackupp*17*

Listing OSTs

To list all the OSTs for the filesystem:

pfe*21*% lfs osts /nobackupp*17*

Viewing Striping Information

To view the striping information for a specific file or directory:

pfe*21*% lfs getstripe *filename*
pfe*21*% lfs getstripe -d *directory_name*

Note: Omitting the -d flag will display striping for all files in the directory.

File Striping

Files on the Lustre filesystems are striped automatically. This means they are transparently divided into chunks that are written or read simultaneously across a set of OSTs within the filesystem. The chunks are distributed among the OSTs using a method that ensures load balancing. Files larger than 100 gigabytes (GB) *must* be striped in order to avoid taking up too much space on any single OST, which might adversely affect the filesystem.

Benefits include:

- Striping allows one or more clients to read/write different parts of the same file at the same time, providing higher I/O bandwidth to the file because the bandwidth is aggregated over the multiple OSTs.
- Striping allows file sizes larger than the size of a single OST.

However, striping small files increases the number of objects in each file, resulting in increased overhead due to network operations

—sometimes creating server contention. This is not likely to cause problems for files larger than 4 megabytes (MB), but in many cases, striping can cause noticeably slower read/write performance for files smaller than 4 MB.

Progressive File Layout

With the deployment of the new Lustre progressive file layout (PFL) feature, different stripe settings for different segments of a file can be configured in such a way that a small stripe count is used for the beginning segment and larger stripe counts are used for latter segments as the file grows. The newer Pleiades Lustre filesystems, /nobackupp[*10-29*], have all been configured with NAS-selected PFL settings.

Important: In most cases, you should rely on these defaults and not manually set file striping yourself. The information in the following sections are useful when using the older, non-PFL filesystems, /nobackupp[*1-2*].

To learn more about filesystems configured with PFL, see Progressive File Layout with SSD and HDD Pools.

Selecting a Stripe Count

The default stripe count on /nobackupp[*1-2*] filesystems is currently 4. The following examples describe circumstances where it is beneficial to change the stripe count to a different number:

- Your program reads a single large input file, where many nodes read or write different parts of the file at the same time. You should stripe this file adequately to prevent all the nodes from reading from the same OST at the same time. This will avoid creating a bottleneck in which your processes try to read from a single set of disks.
- You have other large files. You should restripe large files adequately to keep capacity balanced across the OSTs and maintain consistent performance for all users.
- Your program waits while a large output file is written. You should stripe the large file so that it will write faster, in order to reduce the amount of time the processors are idle.
- Your program periodically writes several small files from each processor. It is not usually necessary to have a stripe count greater than 1 for small files, and they will be randomly distributed across the OSTs to maintain good performance and capacity balance in aggregate.

Setting Stripe Parameters

There are default stripe configurations for each Lustre filesystem. To check the existing stripe settings on your directory, use the `lfs getstripe` command as follows:

```
pfe21% lfs getstripe -d /nobackup/username
```

In most cases, the only tuning you need to do is to change the number of OSTs that your files are written to. To change the number of OSTs, use the `lfs setstripe` command as follows:

```
pfe21% lfs setstripe -c stripe_count dir|filename
```

Note: The stripe settings of an existing file cannot be changed. If you want to change the settings of a file, create a new file with the desired settings and copy the existing file to the newly created file.

Examples of Striping

Newly created files and directories inherit the stripe settings of their parent directories. You can take advantage of this feature by organizing your large and small files into separate directories, then setting a stripe count on the large-file directory so that all new files created in the directory will be automatically striped. For example, to create a directory called "restart" with a stripe count of 8, run:

```
pfe21% mkdir restart
pfe21% lfs setstripe -c 8 restart
```

You can "pre-create" a file as a zero-length striped file by running `lfs setstripe` as part of your job script or as part of the I/O routine in your program. Then, you can write to that file later. For example, to pre-create the file "big_dir.tar" with a stripe count of 20, and then add data from the large directory "big_dir," run:

```
pfe21% lfs setstripe -c 20 big_dir.tar
pfe21% tar cf big_dir.tar big_dir
```

Advanced Parameters

The following stripe parameters are rarely needed, but could potentially be useful, depending on your application I/O:

Stripe size: `-S`
: Sets the size of the chunks in bytes. Use with `k`, `m`, or `g` to specify units of KB, MB, or GB, respectively (for example, `-S 2m`). The specified size must be a multiple of 65,536 bytes (64 KB). The default size is 1 MB for all Pleiades Lustre filesystems; specify `0` to use the default.

Stripe offset: `-o`
: Sets the index of the OST where the first stripe is to be placed. The default is -1, which results in random selection. Using a non-default value is *not* recommended.

See the **lfs man page** for more options and information.

Main Lustre Components

Lustre filesystem components include:

Metadata Server (MDS)
: Service nodes that manage all metadata operations, such as assigning and tracking the names and storage locations of directories and files on the OSTs. There is 1 MDS per filesystem.

Object Storage Target (OST)
: Storage devices where users' file data is stored. The size of each OST varies from approximately 7 terabytes (TB) to approximately 22 TB, depending on the Lustre filesystem. The capacity of a Lustre filesystem is the sum of the sizes of all OSTs. There are multiple OSTs per filesystem.

Metadata Target (MDT)
: A storage device where the metadata (name, ownership, permissions and file type) are stored. There is 1 MDT per filesystem.

Object Storage Server (OSS)
: Service nodes that run the Lustre software stack, provide the actual I/O service and network request handling for the OSTs, and coordinate file locking with the MDS. Each OSS can serve multiple OSTs. The aggregate bandwidth of a Lustre filesystem can approach the sum of bandwidths provided by the OSSes. There can be 1 or multiple OSSes per filesystem.

Lustre Clients
: Compute nodes that mount the Lustre filesystem, and access/use data in the filesystem. There are commonly thousands of Lustre clients per filesystem.

At NAS, Lustre (/nobackup) filesystems are shared among many users and many application processes, which can cause contention for various Lustre resources. This article explains how Lustre I/O works, and provides best practices for improving application performance.

How Does Lustre I/O Work?

When a client (a compute node from your job) needs to create or access a file, the client queries the metadata server (MDS) and the metadata target (MDT) for the layout and location of the file's stripes. Once the file is opened and the client obtains the striping information, the MDS is no longer involved in the file I/O process. The client interacts directly with the object storage servers (OSSes) and object storage targets (OSTs) to perform I/O operations such as locking, disk allocation, storage, and retrieval.

If multiple clients try to read and write the same part of a file at the same time, the Lustre distributed lock manager enforces coherency so that all clients see consistent results.

Jobs being run on Pleiades contend for shared resources in NAS's Lustre filesystem. Each server that is part of a Lustre filesystem can only handle a limited number of I/O requests (read, write, stat, open, close, etc.) per second. An excessive number of such requests, from one or more users and one or more jobs, can lead to contention for storage resources.. Contention slows the performance of your applications and weakens the overall health of the Lustre filesystem. To reduce contention and improve performance, please apply the examples below to your compute jobs while working in our high-end computing environment.

Best Practices

Avoid Using ls -l

The ls -l command displays information such as ownership, permission, and size of all files and directories. The information on ownership and permission metadata is stored on the MDTs. However, the file size metadata is only available from the OSTs. So, the ls -l command issues RPCs to the MDS/MDT and OSSes/OSTs for every file/directory to be listed. RPC requests to the OSSes/OSTs are very costly and can take a long time to complete if there are many files and directories.

- Use ls by itself if you just want to see if a file exists
- Use ls -l *filename* if you want the long listing of a specific file

Avoid Having a Large Number of Files in a Single Directory

Opening a file keeps a lock on the parent directory. When many files in the same directory are to be opened, it creates contention. A better practice is to split a large number of files (in the thousands or more) into multiple subdirectories to minimize contention.

Avoid Accessing Small Files on Lustre Filesystems

Accessing small files on the Lustre filesystem is not efficient. When possible, keep them on an NFS-mounted filesystem (such as your home filesystem on Pleiades /u/*username*) or copy them from Lustre to /tmp on each node at the beginning of the job, and then access them from /tmp.

Keep Copies of Your Source Code on the Pleiades Home Filesystem and/or Lou

Be aware that files under /nobackup are *not* backed up. Make sure that you save copies of your source codes, makefiles, and any other important files on your Pleiades home filesystem. If your Pleiades home directory quota isn't large enough to keep all of these files, you can request a larger quota and/or create tarred copies of these files on Lou.

Avoid Accessing Executables on Lustre Filesystems

There have been a few incidents on Pleiades where users' jobs encountered problems while accessing their executables on the /nobackup filesystem. The main issue is that the Lustre clients can become unmounted temporarily when there is a very high load on the Lustre filesystem. This can cause a bus error when a job tries to bring the next set of instructions from the inaccessible executable into memory.

Executables run slower when run from the Lustre filesystem. It is best to run executables from your home filesystem on Pleiades. On rare occasions, running executables from the Lustre filesystem can cause executables to be corrupted. Avoid copying new executables over existing ones of the same name within the Lustre filesystem. The copy causes a window of time (about 20 minutes) where the executable will not function. Instead, the executable should be accessed from your home filesystem during runtime.

Limit the Number of Processes Performing Parallel I/O

Given that the numbers of OSSes and OSTs on Pleiades are about a hundred or fewer, there will be contention if a large number of processes of an application are involved in parallel I/O. Instead of allowing all processes to do the I/O, choose just a few processes to do the work. For writes, these few processes should collect the data from other processes before the writes. For reads, these few

processes should read the data and then broadcast the data to others.

Understand the Effect of Stripe Counts/Sizes for MPI Collective Writes

For programs that call MPI collective write functions, such as MPI_File_write_all, MPI_File_write_at_all, and MPI_File_write_ordered, it is important to understand the effect of stripe counts on performance.

Background

MPI I/O supports the concept of collective buffering. For some filesystems, when multiple MPI processes are writing to the same file in a coordinated manner, it is much more efficient for the different processes to send their writes to a subset of processes in order to do a smaller number of bigger writes. By default, with collective buffering, the write size is set to be the same as the stripe size of the file.

With Lustre filesystems, there are two main factors in the SGI MPT algorithm that chooses the number of MPI processes to do the writes: the stripe count and number of nodes. When the number of nodes is greater than the stripe count, the number of collective buffering processes is the same as the stripe count. Otherwise, the number of collective buffering processes is the largest integer less than the number of nodes that evenly divides the stripe count. In addition, MPT chooses the first rank from the first *n* nodes to come up with *n* collective buffering processes.

Note: Intel MPI behaves similarly to SGI MPT on Lustre filesystems.

Enabling Collective Buffering Automatically

You can let each MPI implementation enable collective buffering for you, without any code changes.

SGI MPT automatically enables collective buffering for the collective write calls using the algorithm described above. This method requires no changes in the user code or in the mpiexec command line. For example, if the stripe count is 1, only rank 0 does the collective writes, which can result in poor performance. Therefore, experimenting with different stripe counts on the whole directory and/or individual files is strongly recommended.

Intel MPI also does collective buffering, similar to SGI MPT, when the I_MPI_EXTRA_FILESYSTEM and I_MPI_EXTRA_FILESYSTEM_LIST variables are set appropriately, as follows:

```
mpiexec.hydra  -env I_MPI_EXTRA_FILESYSTEM on \
      -env  I_MPI_EXTRA_FILESYSTEM_LIST lustre \
      -np xx a.out
```

Enabling Collective Buffering via Code Changes

In this method, you provide "hints" in the source code to inform MPI what to do with specific files. For example:

```
  call MPI_Info_create(info, status)
  call MPI_Info_set(info, "romio_cb_write", "enable", STATUS)
  call MPI_Info_set(info, "striping_unit", "1048576", STATUS)
  call MPI_Info_set(info, "striping_factor", "16", STATUS)
  ...
  call MPI_File_open(MPI_COMM_WORLD, file_name, MPI_MODE_WRONLY, info, unit, status)
```

Note: The hints are only advisory and may not be honored. For example, SGI MPT 2.12r26 honors these hints, but MPT 2.14r19 does not. Intel MPI 5.0x honors these hints when the I_MPI_EXTRA_FILESYSTEM and I_MPI_EXTRA_FILESYSTEM_LIST variables are set appropriately, as follows:

```
mpiexec.hydra  -env I_MPI_EXTRA_FILESYSTEM on \
      -env  I_MPI_EXTRA_FILESYSTEM_LIST lustre \
      -np xx a.out
```

Stripe Align I/O Requests to Minimize Contention

Stripe aligning means that the processes access files at offsets that correspond to stripe boundaries. This helps to minimize the number of OSTs a process must communicate for each I/O request. It also helps to decrease the probability that multiple processes accessing the same file communicate with the same OST at the same time.

One way to stripe-align a file is to make the stripe size the same as the amount of data in the write operations of the program.

Avoid Repetitive "stat" Operations

Some users have implemented logic in their scripts to test for the existence of certain files. Such tests generate "stat" requests to the Lustre server. When the testing becomes excessive, it creates a significant load on the filesystem. A workaround is to slow down the testing process by adding sleep in the logic. For example, the following user script tests the existence of the files WAIT and STOP to decide what to do next.

```
touch WAIT
 rm STOP
```

```
while ( 0 <= 1  )
 if(-e WAIT) then
   mpiexec ...
   rm WAIT
 endif
 if(-e STOP) then
   exit
 endif
end
```

When neither the WAIT nor STOP file exists, the loop ends up testing for their existence as quickly as possible (on the order of 5,000 times per second). Adding sleep inside the loop slows down the testing.

```
touch WAIT
 rm STOP

 while ( 0 <= 1  )
  if(-e WAIT) then
    mpiexec ...
    rm WAIT
  endif
  if(-e STOP) then
    exit
  endif
  sleep 15
 end
```

Avoid Having Multiple Processes Open the Same File(s) at the Same Time

On Lustre filesystems, if multiple processes try to open the same file(s), some processes will not able to find the file(s) and your job will fail.

The source code can be modified to call the sleep function between I/O operations. This will reduce the occurrence of multiple, simultaneous access attempts to the same file from different processes.

```
100  open(unit,file='filename',IOSTAT=ierr)
     if (ierr.ne.0) then
      ...
     call sleep(1)
     go to 100
     endif
```

When opening a read-only file in Fortran, use ACTION='read' instead of the default ACTION='readwrite'. The former will reduce contention by not locking the file.

```
open(unit,file='filename',ACTION='READ',IOSTAT=ierr)
```

Avoid Repetitive Open/Close Operations

Opening files and closing files incur overhead and repetitive open/close should be avoided.

If you intend to open the files for read only, make sure to use ACTION='READ' in the open statement. If possible, read the files once each and save the results, instead of reading the files repeatedly.

If you intend to write to a file many times during a run, open the file once at the beginning of the run. When all writes are done, close the file at the end of the run.

See Lustre Basics for more information.

Use the Soft Link to Refer to Your Lustre Directory

Your /nobackup directory is created on a specific Lustre filesystem, such as /nobackupp17 or /nobackupp18, but you can use a soft link to refer to the directory no matter which filesystem it is on:

/nobackup/*your_username*

By using the soft link, you can easily access your directory without needing to know the name of the underlying filesystem. Also, you will not need to change your scripts or re-create any symbolic links if a system administrator needs to migrate your data from one Lustre filesystem to another.

Preserve Corrupted Files for Investigation

When you notice a corrupted file in your /nobackup directory, it is important to preserve the file to allow NAS staff to investigate the cause of corruption. To prevent the file from being accidentally overwritten or deleted by your scripts, we recommend that you rename the corrupted file using:

pfe% mv *filename filename*.corrupted

Note: Do not use cp to create a new copy of the corrupted file.

Report the problem to NAS staff by sending an email to support@nas.nasa.gov. Include how, when, where the corrupted file was generated, and anything else that may help with the investigation.

Best Practices for Non-PFL Filesystems

Important: The tips in this section apply only to /nobackupp[*1-2*].

Use a Stripe Count of 1 for Directories with Many Small Files

Note: Skip this tip if you are using the PFL filesystems, nobackupp[*10-29*].

If you must keep small files on Lustre, be aware that stat operations are more efficient if each small file resides in one OST. Create a directory to keep small files in, and set the stripe count to 1 so that only one OST will be needed for each file. This is useful when you extract source and header files (which are usually very small files) from a tarfile. Use the Lustre utility lfs to create a specific striping pattern, or find the striping pattern of existing files.

```
pfe21% mkdir dir_name
pfe21% lfs setstripe -c 1 dir_name
pfe21% cd dir_name
pfe21% tar -xf tar_file
```

If there are large files in the same directory tree, it may be better to allow them to stripe across more than one OST. You can create a new directory with a larger stripe count and copy the larger files to that directory. Note that moving files into that directory with the mv command will not change the stripe count of the files. Files must be *created in* or *copied* to a directory to inherit the stripe count properties of a directory.

```
pfe21% mkdir dir_count_4
pfe21% lfs setstripe -c 4 dir_count_4
pfe21% cp file_count_1 dir_count_4
```

If you have a directory with many small files (less than 100 MB) and a few very large files (greater than 1 GB), then it may be better to create a new subdirectory with a larger stripe count. Store just the large files and create symbolic links to the large files using the symlink command ln.

```
pfe21%  mkdir dir_name
pfe21%  lfs setstripe -c 16 dir_name
pfe21%  ln dir_name/large_file  large_file
```

Increase the Stripe Count for Parallel Access to the Same File

Note: Skip this tip if you are using the PFL filesystems, nobackupp[*10-29*].

The Lustre stripe count sets the number of OSTs the file will be written to. When multiple processes access blocks of data in the same large file in parallel, I/O performance may be improved by setting the stripe count to a larger value. However, if the stripe count is increased unnecessarily, the additional metadata overhead can degrade performance for small files.

By default, the stripe count is set to 4, which is a reasonable compromise for many workloads while still providing efficient metadata access (for example, to support the ls -l command). However, for large files, the stripe count should be increased to improve the aggregate I/O bandwidth by using more OSTs in parallel. In order to achieve load balance among the OSTs, we recommend using a value that is an integral factor of the number of processes performing the parallel I/O. For example, if your application has 64 processes performing the I/O, you could test performance with stripe counts of 8, 16, and 32.

TIP: To determine which number to start with, find the approximate square root of the size of the file in GB, and test performance with the stripe count set to the integral factor closest to that number. For example, for a file size of 300 GB the square root is approximately 17; if your application uses 64 processes, start performance testing with the stripe count set to 16.

Restripe Large Files

Note: Skip this tip if you are using the PFL filesystems, nobackupp[*10-29*].

If you have other large files, make sure they are adequately striped. You can use a minimum of one stripe per 100 GB (one stripe per 10 GB is recommended), up to a maximum stripe count of 120. If you plan to use the file as job input, consider adjusting the stripe count based on the number of parallel processes, as described in the previous section.

If you have files larger than 15 TB, please contact User Services for more guidelines specific to your use case.

We recommend using the shiftc tool to restripe your files. For example:

1. Run ls -lh to view the size of the file(s):

   ```
   % ls -lh data/large_file data/huge_file
   -rw-rw---- 1 zsmith  g1001 555G Apr 14 22:21 data/large_file
   -rw-rw---- 1 zsmith  g1001 3.2T Apr 14 22:21 data/huge_file
   ```

   When a file is less than 1,200 GB, simply use one stripe per 10 GB. For a larger file, you can specify a maximum stripe count of 120.

2. Use shiftc to copy the file to a new file with this number of stripes:

   % shiftc --stripe=10g *large_file large_file*.restripe
   % shiftc --stripe=120 *huge_file huge_file*.restripe

3. Verify that the file was successfully copied. You should receive an email report generated by the shiftc command, or you can run shiftc --status. In the email or status output, check that the state of the operation is "done".

4. Move the new file in place of the old file:

   % mv *large_file*.restripe *large_file*
   % mv *huge_file*.restripe *huge_file*

For more information, see Using Shift for Local Transfers and Tar Operations.

Stripe Files When Moving Them to a Lustre Filesystem

Note: Skip this tip if you are using the PFL filesystems, nobackupp[*10-29*].

When you copy large files onto the Lustre filesystems, such as from Lou or from remote systems, be sure to use a sufficiently increased stripe count. You can do this before you create the files by using the lfs setstripe command, or you can transfer the files using the shiftc tool, which automatically stripes the files.

Note: Use shiftc (instead of tar) when you create or extract tar files on Lustre.

See the following articles for more information:

- Lustre Basics - Setting Stripe Parameters
- Shift Command Options

Reporting Problems

If you report performance problems with a Lustre filesystem, please be sure to include the time, hostname, PBS job number, name of the filesystem, and the path of the directory or file that you are trying to access.Your report will help us correlate issues with recorded performance data to determine the cause of efficiency problems.

Quota Policy on Disk Space and Files

Filesystems on Pleiades and Lou have the following types of quotas:

- Limits on the total disk space occupied by your files
- Limits on the number of files (represented by inodes) you can store, regardless of size. For quota purposes, directories count as files.

Hard and Soft Quota Limits

NAS quotas have hard limits and soft limits. Hard limits should never be exceeded. Soft limits can be exceeded temporarily, for a grace period of 14 days. If your data remains over the soft limit for more than 14 days, the soft limit is enforced as a hard limit. On some filesystems, this means that write attempts will fail. On other filesystems, it will mean that your PBS jobs will no longer start. To reduce your data to below the quota limits, you can delete unneeded files or copy important files elsewhere, such as the Lou mass storage system, and then remove them locally.

Filesystem quotas are shown in the following table:

	Pleiades	**Lou**
$HOME	NFS	XFS
Space: soft	8 GB	none
Space: hard	10 GB	none
Inode: soft	none	250,000
Inode: hard	none	300,000
/nobackup	Lustre /nobackupp*X*	N/A
Space: soft	1 TB	N/A
Space: hard	2 TB	N/A
Inode: soft	500,000	N/A
Inode: hard	600,000	N/A

To learn how to check your disk space, inode usage, and quotas, see the following articles:

- Pleiades Home Filesystem
- Pleiades Lustre Filesystems
- The Lou Mass Storage System

Email Warnings and Consequences

It is expected that your data will exceed your soft limits as needed. When this occurs, you will begin to receive daily emails to inform you of your current disk space and how much of your grace period remains. However, if your data is still over the soft limit or if you reach the hard limit, restrictions are put in place.

The following table shows the quota enforcement policy for each type of filesystem.

Filesystem	**Grace Period**	**Enforcement**
/nobackupp*X*	2 weeks	Batch jobs won't run
/nobackupnfs2	2 weeks	Writes are disabled
Pleiades home directory	2 weeks	Writes are disabled
Lou home directory	2 weeks	Writes are disabled

More details are available in the following sections.

Lustre Filesystems (/nobackupp*X*)

If your grace period has expired, then your batch queue access is restricted and no new jobs can be run until your data on the Lustre filesystem is reduced to an amount below the soft limit. Any jobs that are running will continue to run. If you reach the hard limit, your batch access is immediately restricted, but any running jobs will continue.

Pleiades Home Directory and /nobackupnfs2

If your grace period has expired, then any writes to that filesystem will fail, and any jobs that attempt to write to the filesystem will fail. You will be unable to write to the filesystem until your data is reduced to an amount below the soft limit. Similarly, if you reach the hard limit, any writes will immediately fail and you will be unable to write to the filesystem until your data is reduced to an

amount below the hard limit.

Lou Home Directory

If your grace period has expired, then any writes that attempt to create new files will fail. You will be unable to create any new files until the number of files is reduced to below the soft limit. If you reach the hard limit, any writes that attempt to create new files will fail until the number of files is reduced to below the hard limit.

The maximum size of a tar file moved to Lou should not exceed 2 TB. If you need to archive larger files, please contact the NAS Control Room at support@nas.nasa.gov for assistance.

Note: The Shift tool, which is a convenient way to transfer files to Lou, can created a set of smaller independent tar files from a single large directory. For more information, see Shift File Transfer Overview.

Changing Your Quotas

If an account needs larger quota limits, send an email justification tosupport@nas.nasa.gov. Your request will be reviewed by management for approval.

The home filesystems on Pleiades are HPE/SGI NEXIS 9000 filesystems that are NFS-mounted on all of the Pleiades front-end (PFE) nodes and compute nodes. Each NAS user is provided a home directory on the Pleiades home filesystems, which are the home filesystems for Pleiades, Aitken, Electra, and Endeavour. After you are granted an account, your home directory is set up automatically during your first login.

Your home directory has a quota of 8-10 GB of storage. For temporary storage of larger files, use the Lustre /nobackup filesystems. For long-term storage, use the Lou mass storage system.

Quota Limits and Policy

Disk space quota limits are enforced on the Pleiades home filesystems. By default, the soft limit is 8 GB and the hard limit is 10 GB. There are no inode limits on the home filesystem.

To check your quota and usage on your home filesystem, run the `quota -v` command as follows:

```
%quota -v
Disk quotas for user username (uid xxxx):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
saturn-ib1-0:/mnt/home2
             7380152  8000000 10000000          190950       0       0
```

The NAS quota policy states that if you exceed the soft quota (the number listed under `quota` in the above sample output), an email will be sent to inform you of your current usage and how much of the grace period remains. It is expected that you will occasionally exceed your soft limit; however, after 14 days, any attempts to write to the filesystem will result in error. Any attempt to exceed the hard limit (the number listed under `limit` in the above sample output) will result in error.

If you believe that you have a long-term need for higher quota limits on the Pleiades home filesystem, send an email justification to support@nas.nasa.gov. Your request will be reviewed by the HECC Deputy Project Manager for approval.

Note: For temporary storage of larger files, or a large number of files, use your Lustre /nobackupp*X* directory. For normal long-term file storage, transfer your files to the Lou mass storage systems.

See also: Quota Policy on Disk Space and Files.

TIP: If you receive the following error message when logging in, you won't be able to run X applications. This error is usually due to exceeding your home filesystem quota. To avoid the error, decrease your disk usage.

```
/usr/X11R6/bin/xauth: error in locking authority file /u/username/.Xauthority
```

Backup Schedule

Files on the home filesystem are backed up daily.

Pleiades, Aitken, Electra, and Endeavour share several filesystems intended to provide working space for compute jobs. These filesystems, called "nobackup" (/nobackupp*X*), provide over 90 petabytes (PB) of disk space and serve many thousands of cores. The /nobackup filesystems are Lustre-based filesystems managed under Lustre software version 2.*x*.

Using the Lustre /nobackup Filesystems

As the name suggests, these "nobackup" filesystems are for temporary use, and are not backed up. Lustre can handle many large files, but you cannot use the Lustre filesystems for long-term storage. If you want to save your files, move them to Lou.

Each Lustre filesystem is shared among many users. To learn how to achieve good I/O performance for your applications and avoid impeding the I/O operations of other users, read the related articles listed at the bottom of the page.

Lustre filesystem configurations are summarized at the end of this article.

WARNING: Any files that are removed from Lustre /nobackup filesystems *cannot* be restored. You should store essential data on the Lou mass storage system (lfe[*5-8*]) or on other, more permanent storage.

Which Lustre Filesystem is Assigned to Me?

Once you are granted a NAS account, one of the Lustre filesystems will be assigned to you. To find out which one, run the ls command as follows:

pfe*21*% ls -l /nobackup/*your_username*

In the output, look for the filesystem name (nobackupp*X*). For example, the following output shows that the user's assigned filesystem is /nobackupp17:

lrwxrwxrwx 1 root root 19 Sep 23 2021 /nobackup/*your_username* -> /**nobackupp17**/*your_username*

Default Quota and Policy on /nobackup Filesystems

Disk space and inode quotas are enforced on the /nobackup filesystems. The default soft and hard quota limits for inodes are 500,000 and 600,000, respectively. Quotas for the disk space are 1 terabyte and 2 terabytes, respectively. To check your disk space, inode usage, and quota on your /nobackup filesystem, run the lfs command as follows:

```
% lfs quota -h -u username /nobackup/username
Disk quotas for user username (uid nnnn):
     Filesystem    used   quota   limit   grace   files   quota   limit   grace
/nobackup/username      4k     1.024T  2.048T      -       1   500000  600000      -
```

It is expected that your data will occasionally exceed the soft limit. However, after a 14-day grace period, your access to the batch queues will be restricted and you will not be able to run new jobs until your data is reduced below the soft limit. If you reach the hard limit, your batch access will immediately be restricted, but any running jobs will continue.

If you anticipate a long-term need for higher quota limits, please send a justification via email to support@nas.nasa.gov. Your request will be reviewed by the HECC Deputy Project Manager for approval.

For more information, see Quota Policy on Disk Space and Files.

Note: When a Lustre filesystem is full, the jobs writing to it will hang. A Lustre error with code -28 in the system log file indicates that the filesystem is full. NAS support staff will typically send emails to those using the most space, with a request to clean up their files.

Lustre Filesystem Configurations

The way your Lustre filesystem is configured determines how your files are striped. There are two types of configurations: Progressive File Layout (PFL), and the standard Lustre configuration. Most of the /nobackup filesystems have PFL configurations.

Progressive File Layout Configurations

Lustre's Progressive File Layout (PFL) feature enables filesystems to dynamically change the stripe count for files based on their size. This means that the files are dynamically striped, therefore, you should ***not*** set custom stripe size or stripe counts.

The Lustre PFL filesystems are /nobackupp[*10-29*]; among them, /nobackupp[*17-19, 27-29*] are equipped with a small number of solid state drives (SSDs) in addition to the hard disk drives (HDDs), while /nobackupp[*10-13,15-16*] only have HDDs.

The HDD configurations of /nobackupp[*10-29*] are listed in the table below with the abbreviation nbp*X*. P = petabytes; T = terabytes; M = megabytes.

Lustre PFL HDD Configurations

Filesystem	nbp10	nbp11	nbp12,13,15	nbp17,18	nbp19	nbp27,28	nbp29
# of OSTs	46	69	23	32	16	32	16

size/OST	70.7 T	70.7 T	70.7 T	565 T	565 T	565 T	565 T
Total Space	3.2 P	4.8 P	1.6 P	18.6 P	9.3 P	18.6 P	9.3 P

For /nobackupp[*10-13,15-16*], which have only HDDs, a common default PFL stripe setting is used as follows:

Lustre PFL Stripe Counts per File Size

File Size	0 - 10 MB	10 MB - 17 GB	17 - 68 GB	> 68 GB
Stripe Count	1	4	8	16

For /nobackupp[*17-19, 27-29*], different PFL configurations are set among the SSD and HDD pools. To learn more, see Progressive File Layout with SSD and HDD Pools.

Standard Configurations

The standard Lustre filesystems are /nobackupp1 and 2; in the table below, these are abbreviated as nbp*X*. P = petabytes; T = terabytes; M = megabytes.

Standard Lustre Configurations

Filesystem	nbp1	nbp2
# of OSTs	144	342
size/OST	43.6/57.2 T	43.6/71.2 T
Total Space	7.1 P	18 P
Default Stripe Size	1 M	1 M
Default Stripe Count	4	4

Backing Up Data in the Lou Mass Storage System

For safe, long-term data storage, transfer your files to the Lou mass storage system, which allows you to retrieve your stored files quickly and securely whenever you need them.

The Lou mass storage system is an Intel Xeon Gold 6154 "Skylake"-based cluster combined with a Spectra Logic tape storage archive. Lou uses a parallel Data Migration Facility (DMF) system to provide high speed and bandwidth for data transfers between Lou's disks and tapes.

Connecting to Lou

Lou's four hosts, known as Lou front-end systems (LFEs), are designated as lfe[*5-8*].

You can connect automatically to the LFE that has the lowest load by using the "lou" or "lfe" hostnames in yourssh command line: ssh lfe or ssh lou. This behavior is similar to the load balancing process that occurs on the Pleiades front-end nodes (PFEs) when you run ssh pfe.

Transferring Files Between /nobackup Filesystems and Lou

The Lustre (/nobackup) filesystems are mounted on the LFEs. To transfer files between your /nobackup filesystem and your Lou home filesystem, use the local file transfer commands shiftc, cp, or mcp. For example:

lfe% cp /nobackup/*your_username*/filename /u/*your_username*

You can also create a tar file that contains the data in one of your /nobackup subdirectories and then store the tar file in the LFE home filesystem. For example:

lfe% cd /nobackup/*your_username*
lfe% tar cf /u/*your_username*/*mydir*.tar *mydir*

Transferring Files from Your Local System to Lou

Recommended: Use the Shift tool (shiftc), which accepts both the alias and specific hostnames: lou, lfe, or lfe[*5-8*].

For more information about transferring files to and from your local system, see Remote File Transfer Commands.

Data Migration Between Disk and Tapes

In addition to 7.6 petabytes of disk space, Lou has 51 LTO-8 tape drives. Each LTO-8 tape holds 12 TB of uncompressed data, for a total storage capacity of approximately 1040 petabytes, or one exabyte (with normal 35% compression). Data migration (from disk to tape) and unmigration (from tape to disk) are managed by the Data Migration Facility (DMF).

Data stored on Lou's home filesystems (on disk) are automatically migrated to tape. Two copies of your data are written to tape media in silos located in separate buildings. When it is necessary to make room for more data, some files that have been written to tape may have their data "released" from disk. This means that the file is still visible on the filesystem, but the data must be retrieved from tape before the file can be used.

When a file completes writing to tape on Lou, its ctime attribute is updated to signal that it is available to be backed up by system software. This should not have an affect on any file transfer applications such as rsync.

If you need to retrieve data that is on tape, be sure to unmigrate the data from tape to your home filesystem on Lou before transferring it to other systems.

TIP: If you use the Shift tool (shiftc) for file transfers, it will automatically ensure that files on Lou are online before the transfer.

If you are not using Shift, you can use the following DMF commands to retrieve your files from tape:

$ dmls -al *file1 file2 . . .* # show the status of your files.
$ dmget *file1 file2 . . .* & # retrieve your file from tape.

At this point, you can start your transfer and the files will transfer as they come online.

WARNING: Do not use the /nobackup filesystems for long-term data storage. As the names suggest, these filesystems are not backed up, so any files that are removed cannot be restored. You should store essential data on more permanent storage devices, such as Lou.

For more tips on how to use the Lou storage systems more effectively, see:

- Portable File Names and Sizes
- Dealing with Slow File Retrieval

Post-Processing Your Data

To perform post-processing tasks on Lou data, use the Lou data analysis nodes, which provide dedicated PBS resources for that purpose.

Post-processing is not permitted on the LFEs. To enforce this, a system monitoring tool called query_wms runs on the LFEs and kills any user process that uses more than 1 GB of memory.

Quota Limits On Lou

There are no disk quota limits on your Lou home filesystem. However, there *are* limits on the number of files (inodes):

- 250,000 inode soft limit (14-day grace period)
- 300,000 inode hard limit

To check your quota status and usage on your home filesystem, log into Lou (lfe) and run the quota -ls command as follows:

```
lfe% quota -ls
Disk quotas for user username (uid xxxx):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
/dev/cxvm/sfa2_s2n
               87422M       0       0           59381    250k    300k
```

See Quota Policy on Disk Space and Files for more information about HECC system quotas.

Storing Data in Pleiades Home Filesystems

All data stored in your home filesystem of any NAS system, such as Pleiades, are automatically backed up *daily* under script control.

If you lose important data from a home filesystem and need to have it restored, please send a request to the NAS Control Room at support@nas.nasa.gov and provide the following information:

- System name
- Your username
- The directory and/or file name(s) that you wish to restore
- The date the data was last touched

Note: Disk space quota limits are imposed on the Pleiades home filesystems. If your data exceeds your disk space quota limits, reduce your usage by deleting unneeded files, copying important files to Lou, or copying them to storage space at your local site and then removing them from Pleiades.

Storing Data in Pleiades /nobackup Filesystems

Data in any /nobackup filesystem of any NAS system, such as Pleiades /nobackupp*X*, are *not* backed up by NAS. You are responsible for performing your own backups.

Note: Disk space and inode quota limits are imposed on the Pleiades Lustre filesystems (/nobackupp*X*). If your data exceeds the soft quota limits, reduce your usage by removing some files and/or archiving any important data on Lou or on the storage space at your local site.

In addition, when the overall usage of a /nobackup filesystem is near its capacity, files can be deleted by system administrators. Normally, the few top users are sent emails and asked to clean up their disk space.

Using Shift to Copy Files to Lou

The Pleiades /nobackup filesystems (for example, /nobackup[*1-8*]) are mounted on Lou. As a result, you can log into Lou and copy the files from your /nobackup filesystem to your home directory on Lou.

We recommend using the Shift tool (shiftc), which will automatically utilize the highest performing local file transport. You can also use shiftc if you need to initiate the transfer from Pleiades. If you need to encrypt the data, even within the NAS enclave, then use the shiftc --encrypt option.

The Data Migration Facility (DMF) keeps disk space on the Lou mass storage systems available to users by moving unused files to storage on tape. Use the DMF commands to list, put, find, and get files on tape, with options for controlling the file transfers.

At the NAS facility, the Lou filesystems support a virtual storage manager feature called the Data Migration Facility (DMF). Its purpose is to allow users to keep an increased volume of data in the files under their home directories by migrating those files that are not currently in use to tape storage, thereby allowing active disk space to be available for active files.

Migrated files are retrieved to active disk when you attempt to read or write to them, and the whole process is substantially transparent, aside from a possible time lag while a file is being retrieved. Using the NAS-developed tool, Shift, you can pre-transfer files from tape to avoid the time lag during a job.

The process of migrating files by backing them up from disk to another medium is implemented using Spectra Logic tape libraries.

DMF Commands

User commands associated with DMF are below, followed by usage examples:

dmls	Directory listing showing file migration status
dmget	Retrieve migrated files to disk
dmput	Cause files to migrate to backup on tape
dmfind	Find files under a directory hierarchy
dmcopy	Copy all or part of offline files
dmattr	List attributes of files

The dmls Command

The dmls command is much like the standard ls command for file or directory listings, with the addition of an option for displaying the DMF status of files. Actually, dmls is derived from the GNU ls command, so it has a few extra "bells and whistles" compared to the standard ls command; for details, see the **dmls man page**.

Example showing the extra status field:

```
% ls -l
total 64
-rw-r-----   1 aeneuman madmag    14713 Mar  1 17:02 file1
-rw-r-----   1 aeneuman madmag    17564 Mar  1 17:02 file2

% dmls -l
total 33
-rw-r-----  1 aeneuman madmag    14713 Mar  1 17:02 (REG) file1
-rw-r-----  1 aeneuman madmag    17564 Mar  1 17:02 (REG) file2
```

DMF has several possible states for files. The first three shown below are the most likely to appear:

REG	Regular. The file exists only online, on active disk.
OFL	Offline. The file's directory entry remains on disk, but its data blocks are located offline only (on tape).
DUL	Dual-state. Identical copies of the file exist online (on disk) and offline (on tape). The online copy will persist if there is no demand for free space in its filesystem. When free space is needed, the online copy of the file is removed, leaving just the offline copy; the file becomes "offline." If you make any change to a dual-state file, the offline copy becomes out of date and invalid, and the file is once again a "regular" file.
MIG	Migrating. The file is in process of migrating from disk to tape.
UNM	Unmigrating. The file has been recalled and is in process of moving back from tape to disk.
NMG	Nonmigratable. The file cannot be migrated.
INV	Invalid. DMF cannot determine the file's state. The most likely reason is that it is in a filesystem that does not use DMF.

The dmget Command

The dmget command explicitly requests an offline file to be retrieved to disk. This command is not strictly required in order to retrieve offline files; any program that tries to use the file will cause it to be retrieved first.

The following example shows that referencing an offline file retrieves it after a pause. Notice that file C was "offlined." Using the command file C caused this file to be retrieved and its status changed from OFL to DUL.

```
% dmls -l
total 1404
-rw-r-----   1 aeneuman madmag     20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag    201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag   1209300 Mar  3 11:20 (OFL) C

% file B
```

```
[The response is immediate, because the file is already online:]
B:          English text

% file C
[There will be a short delay before the response, while the file is retrieved from lou:]
C:          English text

% dmls -l
total 1404
-rw-r-----   1 aeneuman madmag     20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag    201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag   1209300 Mar  3 11:20 (DUL) C
```

You can retrieve the file explicitly with the command dmget. For example:

```
% dmls -l
total 220
-rw-r-----   1 aeneuman madmag     20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag    201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag   1209300 Mar  3 11:20 (OFL) C

% dmget C ; echo Done
[There will be a short delay before the response, while the file is retrieved from lou:]
Done
```

One motivation for retrieving files explicitly with dmget is that you may be able to save time. If you are working with several files that have been migrated at the same time and reside on the same offline tape cartridge, a dmget command that names all the files would be able to retrieve them all in a single tape mount.

By contrast, simply using the files one after another would cause the tape robot to fetch the tape cartridge, retrieve the first file and put away the tape, then go back and fetch the cartridge, retrieve the second file, put away the tape, and so on.

Another good reason for retrieving files explicitly with dmget is that you control when the retrieval takes place. For example, suppose you have a large, multiprocessor application that is going to read a currently migrated data file. You would prefer the retrieval process to take place with just your shell running, not when your main application has spawned several dozen processes on several dozen nodes, with all of them having to sit idle waiting for the file to be retrieved.

See the dmfind example for an efficient method to recall all files in a directory and its subdirectories.

The dmput Command

In filesystems that are under DMF automated space management, large files that have not been used for awhile will be migrated offline automatically. The definitions of "large" and "awhile" are established on each system by its system administrator.

You can invoke migration explicitly with the dmput command. Useful options are:

-r
: Causes DMF to release a file's online disk space immediately, giving it Offline status. Otherwise, the file would be in Dual status until its disk space was needed for other files.

-w
: Causes the dmput command to wait until migration has completed before returning control. Otherwise the dmput command returns immediately with the file in Migrating status.

Here's an example showing immediate return and changing state:

```
% dmls -l
total 1404
-rw-r-----   1 aeneuman madmag     20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag    201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag   1209300 Mar  3 12:43 (REG) C

% dmput C ; dmls -l
[The immediate response shows that the file is migrating:]
total 1404
-rw-r-----   1 aeneuman madmag     20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag    201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag   1209300 Mar  3 12:43 (MIG) C

[After a delay, the migration to lou is complete:]

% dmls -l
total 1404
-rw-r-----   1 aeneuman madmag     20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag    201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag   1209300 Mar  3 12:43 (DUL) C
```

This example shows delayed return, and release of disk space:

```
% dmls -l
total 1404
-rw-r-----   1 aeneuman madmag     20155 Mar  2 11:24 (REG) A
```

```
-rw-r-----   1 aeneuman madmag     201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag    1209300 Mar  3 15:17 (REG) C

% dmput -r -w C ; dmls -l
[There will be a delay in response, while the file is migrated to lou:]
total 220
-rw-r-----   1 aeneuman madmag      20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag     201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag    1209300 Mar  3 15:17 (OFL) C
```

The dmfind Command

The dmfind command is based on the GNU version of the find command and adds the ability to search for files in a given DMF state. This is handy for determining which files are offline and, therefore, candidates for retrieval with dmget.

Example:

```
% dmls -l
total 220
-rw-r-----   1 aeneuman madmag      20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag     201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag    1209300 Mar  3 15:17 (OFL) C

% dmfind . -state ofl  -print
./C

% dmfind . -state ofl  -print | dmget

[After a short delay, the retrieval from lou is completed:]

% dmls -l
total 1404
-rw-r-----   1 aeneuman madmag      20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag     201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag    1209300 Mar  3 15:17 (DUL) C
```

To efficiently recall all files in a directory named *mydir* and its subdirectories, use the following command:

```
% dmfind mydir -state mig -or -state ofl -print | dmget
```

The dmcopy Command

The dmcopy command copies all or part of an offline file to a destination file, keeping the file offline.

When the offline file is being copied in its entirety, the only arguments needed are the two filenames.

Example:

```
% dmls -l
total 224
-rw-r-----   1 aeneuman madmag      20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag     201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag    1209300 Mar  3 15:17 (OFL) C

% dmcopy  C  newC

[After a short delay, the copy is completed:]

% dmls -l
total 1404
-rw-r-----   1 aeneuman madmag      20155 Mar  2 11:24 (REG) A
-rw-r-----   1 aeneuman madmag     201550 Mar  2 11:24 (REG) B
-rw-r-----   1 aeneuman madmag    1209300 Mar  3 15:17 (OFL) C
-rw-------   1 aeneuman madmag    1209300 Mar  4 15:06 (REG) newC
```

The dmcopy command has options that allow copying just part of the offline file, and specifying offset locations in the source and destination files.

-l *length*
: Specifies a data length to copy in bytes. The default is the whole size of the source file.

-o *source-offset*
: Specifies a byte offset in the offline source file where reading is to begin. The default is zero; that is, reading begins at the start of the source file.

-d *destination-offset*
: Specifies a byte offset in the destination file where writing is to begin. Any bytes in the destination file before this offset are zero filled. The default destination offset is whatever the source offset is.

Example: Skip two records of the offline source file and copy the next three records. Records are 2,048 bytes long.

```
% set RECDLEN=2048
```

```
% set NRECD=3
% @ LENGTH = $NRECD * $RECDLEN
% set NSKIP=2
% @ OFFSET = $NSKIP * $RECDLEN

% dmcopy  -l $LENGTH  -o $OFFSET  -d 0  C  newC

[After a short delay, the copy is completed:]

% dmls -l *C
-rw-r-----   1 aeneuman madmag    1209300 Mar  3 15:17 (OFL) C
-rw-------   1 aeneuman madmag       6144 Mar  4 16:10 (REG) newC
```

Note: If the source file is a regular file, without an offline copy in DMF, the destination file is always zero length. The modes on the source file are ignored when creating the destination file. The modes on the destination file are 0666 (readable and writable by everyone), except where blocked by the current umask. Review your output files' permissions and reset them with the command chmod, as needed.

The dmattr Command

The dmattr command prints selected attributes of specified files. This is most useful in shell scripts.

Some options include:

-a *attr1*,*attr2*,...
: Selects a subset of the reportable attributes.

-d *delimiter*
: Specifies a one-character separator between adjacent values. A space is the default.

-l
: Prints the attributes, one per line, with labels.

Examples for a file named "A"

```
% dmattr -l A
      bfid : 0
     emask : 0
   fhandle : 01000000000000188dede3a4b319c5b9000e000000000001000000000b3fd163
     flags : 0
     nregn : 0
     owner : 4771
      path : A
      size : 20155
     space : 20480
     state : REG

% dmattr A
0 0 01000000000000188dede3a4b319c5b9000e000000000001000000000b3fd163 0 0
4771 A 20155 20480 REG

% dmattr  -a owner,state  -d :  A
4771:REG

% foreach file ( * )
? if (`dmattr -a state $file` == OFL) then
? echo $file is offline
? endif
? end

C is offline
```

Portable File Names

Use portable file names. A name is portable if it contains only ASCII letters and digits, `.', `_', and `-'. Do not use spaces or wildcard characters or start with a `-' or `//', or contain `/-'. Avoid deep directory nesting.

If you intend to have tar archives to be read under MSDOS, you should not rely on case distinction for file names, and you might use the GNU doschk program for helping you further diagnose illegal MSDOS names, which are even more limited than Unix like operating system.

Portable File Sizes

Even though Lou's archive filesystem will allow a file size to be greater than several hundred gigabytes, not all operating systems or filesystems can manage this much data in a single file. If you plan to transfer files to an old Mac or PC desktop you may want to verify the maximum filesize it will support. Likely a single file will need to be less than 4 GB before it will transfer successfully.

Verifying Files Transferred to the Lou Mass Storage System

It is a good practice to confirm whether files are copied correctly to the Lou mass storage system after you transfer them.

TIP: The easiest way to verify the integrity of your file contents is to use the NAS-developed Shift tool to transfer the files. By default, Shift automatically performs a checksum operation on the data at both the source and the destination, as part of the transfer. If corruption is detected, partial file transfers and checksum operations will be performed until the problem is fixed.

If you transfer files using a command other than Shift, such as scp, the simplest and most lightweight method to verify transferred files is to compare the size (disk usage) and number of files of the original with that of the copy. For example, if you use scp to transfer *dir1* from pfe21 to lfe5:

```
pfe21% du -sk dir1
353760  dir1
pfe21% find dir1 | wc -l
51

pfe21% scp -rp dir1 lfe5:

lfe5% du -sk --apparent-size dir1
353684  dir1
lfe5% find dir1 | wc -l
51
```

In the example, the directory sizes nearly match (353760; 353684), and the number of files matches exactly (51).

Note: In most cases the sizes will not match exactly. Using the--apparent-size option of the du command is necessary on the Lou systems because the data may reside on either tape or disk.

Optimizing/Troubleshooting

Increasing File Transfer Rates

If you are moving large files, use the `shiftc` command instead of `cp` or `scp`. An online NAS service can help diagnose your remote network connection issues, and our network experts can work with your specific file transfer problems.

For fastest file transfer between Pleiades /nobackup and Lou, log into Lou and use `shiftc`, `cxfscp`, or `mcp`. A simple `cp` or `tar` will also work, but at slower speeds.

Moving large amounts of data efficiently to or from NAS across the network can be challenging. Often, minor system, software, or network configuration changes can increase network performance an order of magnitude or more.

If you are experiencing slow transfer rates, try these quick tips:

- Pleiades /nobackup are mounted on Lou, enabling disk-to-disk copying, which should give the highest transfer rates. You can use the `shiftc`, `cp`, or `mcp` commands to copy files or even make tar files directly from Pleiades /nobackup to your Lou home directory.
- If using the `scp` command, make sure you are using OpenSSH version 5 or later. Older versions of SSH have a hard limit on transfer rates and are not designed for WAN transfers. You can check your version of SSH by running the command `ssh -V`.
- For large files that are a gigabyte or larger, we recommend using `shiftc`. This application allows for transferring simultaneous streams of data and, when used without the `--secure` option, doesn't have the overhead associated with encrypting all the data (authentication is still encrypted).
- Another reliable option for large file transfers is through the Shift transfer tool, which includes options specific to the NAS environment, such as checking to see whether files residing on Lou are also on tape.

One-on-One Help

If you would like further assistance, contact the NAS Control Room at support@nas.nasa.gov, and a network expert will work with you or your local administrator one-on-one to identify methods for increasing your transfer rates.

To learn about other network-related support areas see End-to-End Networking Services.

Streamlining PBS Job File Transfers to Lou

Some users prefer to streamline the storage of files (created during a job run) to Lou, within a PBS job. Because direct access to the Lou storage nodes from the Pleiades compute nodes and from Endeavour has been disabled, all file transfers to Lou within a PBS job must first go through one of the Pleiades front-end systems (PFEs).

Here is an example of what you can add to your PBS script to accomplish this:

1. ssh to a PFE (for example, pfe*21*) and create a directory on lou where the files are to be copied.

   ssh -q pfe*21* "ssh -q lou mkdir -p $SAVDIR"

   Here, $SAVDIR is assumed to have been defined earlier in the PBS script. Note the use of -q for quiet-mode, and double quotes so that shell variables are expanded prior to the ssh command being issued.

2. Use scp via a PFE to transfer the files.

   ssh -q pfe*21* "scp -q $RUNDIR/* lou:$SAVDIR"

   Here, $RUNDIR is assumed to have been defined earlier in the PBS script.

File Transfers Tips

The following quick and easy techniques may improve your performance rates when transferring files remotely to or from NAS.

This can increase your transfer rates by 5x, compared to older methods such as 3des.

- Transfer files from the /nobackup filesystem, which is often faster than the locally mounted disks.
- If you are using scp and your data is compressible, try adding the -C option to enable file compression, which can sometimes double your performance rates:

  % scp -C *filename user@remote_host.com*:

- For SCP transfers, use a low-process-overhead cipher such as aes128-gcm@openssh.com or arcfour:

  % scp -c -aes128-gcm@openssh.com *filename user@remote_host.com*:

- If you are transferring files from Lou, make sure they are online, rather than on the tape archive, before you perform the transfer operation.

  Note: If you use the shiftc command to transfer your files, it will automatically bring any files that are on the tape archive online before it transfers them. If you are not using shiftc, use the following DMF commands to determine the location of your files and bring them online if necessary:

  % dmls -al *filename* # show the status of your file.
  % dmget *filename* # retrieve your file from tape prior to transferring.

  For a full list of DMF commands, see DMF commands.

- If you are transferring many small files, try using the tar command to compress them into a single file prior to transfer. Copying one large file is faster than transferring many small files.
- For files larger than a gigabyte, we recommended using the Shift Transfer Tool, which can achieve much faster rates than single-stream applications such as scp or rsync.

To improve your performance by modifying your system, see TCP Performance Tuning for WAN Transfers.

If you continue experiencing slow transfers and want to work with a network engineer to help improve file transfers, please contact the NAS Control Room at support@nas.nasa.gov.